ISSN 2710-1673. Artificial Intelligence, 2020, № 3

UDC 004.93 DOI: https://doi.org/10.15407/jai2020.03.007

V.V. Rud
Dnipro University of Technology, Ukraine
19, Dmytra Yavornytskoho St., Dnipro, 49005

ROBOTIC GRASPING WITH OBSTACLE AVOIDANCE USING OCTREES

В.В. Рудь
Національний технічний університет «Дніпровська політехніка», Україна
вул. Дмитра Яворницького, 19, м. Дніпро, 49005

РОБОТИЧНЕ ХАПАННЯ З УНИКНЕННЯМ ПЕРЕШКОД ЗА ДОПОМОГОЮ ДЕРЕВ ОКТАНТІВ

Abstract. This paper considers the problems of the integration of independent manipulator control systems. Areas of control of the manipulator are: recognition of objects and obstacles, identification of objects to be grasped, determination of reliable positions by the grasping device, planning of movement of the manipulator to certain positions with avoidance of obstacles, and recognition of slipping or determination of reliable grasping. This issue is a current problem primarily in industry, general-purpose robots, and experimental robots. This paper considers current publications that address these issues. Existing algorithms and approaches have been found in the management of both parts of the robot manipulator and solutions that combine several areas, or the integration of several existing approaches. There is a brief review of current literature and publications on the above algorithms and approaches. The advantages and disadvantages of the considered methods and approaches are determined. There are solutions that cover either some areas or only one of them, which does not meet the requirements of the problem. Using existing approaches, integration points of existing implementations are identified to get the best results. In the process, a system was developed that analyzes the environment, finds obstacles, objects for interaction, poses for grasping, plans the movement of the manipulator to a specific position, and ensures reliable grasping of the object. The next step was to test the system, test the performance, and adjust the parameters for the best results. The resulting system was developed by the research team of RT-Lions, Technik University, Reutlingen. The hardware research robot includes an Intel Realsense camera, a Sawyer Arm manipulator from Rethink Robotics, and an internally grabbing device.

Keywords: ROS; grasping; object detection; obstacle avoidance; octree; point cloud; motion planning.

Анотація. Дана робота розглядає проблеми інтеграції незалежних систем керування маніпулятором. Областями керування маніпулятором є: розпізнання об’єктів та перешкод, визначення об’єктів для хапання, визначення надійних позицій пристроєм хапання, планування руху маніпулятора до визначених позицій з уникненням перешкод та розпізнання проковзування або визначення надійного хапання. Розглянуте питання є сучасною проблемою, перш за все у галузях промисловості, роботів загального призначення та експериментальних роботів. У даній роботі розглянуто актуальні публікації, що розглядають зазначені питання. Знайдено наявні алгоритми та підходи в управлінні як частинами робота-маніпулятора, так і рішення, що об’єднують у собі декілька областей, або інтеграції між собою декількох існуючих підходів. Є короткий огляд актуальної літератури та публікацій на тему вищезгаданих алгоритмів та підходів. Визначено переваги та недоліки розглянутих методів та підходів. Існують рішення, що покривають або деякі області, або тільки одну з них, що не задовольняє вимогам вирішення проблеми. Користуючись існуючими підходами, визначено точки інтеграції існуючих реалізацій для отримання найкращого результату. У процесі роботи було розроблено систему, що аналізує навколишнє середовище, знаходить перешкоди, об’єкти для взаємодії, пози для хапання, планує рух маніпулятора до визначеної пози та впевнюється у надійному хапанні об’єкта. Наступним кроком стало тестування системи, перевірка працездатності підходу та налаштування параметрів для найкращих результатів. Система, створена в результаті роботи, була розгорнута на дослідницькому роботі команди RT-Lions факультету Technik Університету Ройтлінгена. Апаратнодослідницький робот включає в себе камеру Intel Realsense, маніпулятор Sawyer Arm від Rethink Robotics та пристрій хапання внутрішньої розробки.

Ключові слова: ROS; хапання; розпізнавання об’єктів; уникнення перешкод; дерево октантів; хмара точок

Introduction

Controlling a robot arm is one of the most important parts of robotic software development. It consists of next problems: object recognition, grasp pose selection, obstacle detection, motion planning.

There are existing researches and developments for the mentioned problems. But

combining all existing solutions and approaches might be a more complex task. The final solution should combine the most appropriate solution in each area and bring them together as a complete system.

Problem Statement

There are several areas in object manipulation: grasping, motion planning, slip detection. Communication between them is not established.

Analysis of recent research and publications

The researched problem is relevant in robotic arm development. It was found some studies and literature mention the stated problem. But almost all of them are concentrated on independent solutions while the problem is the integration of small target solutions in a complete system.

There are several basic approaches to control the arm. Including MoveIt! [2], point-cloud for grasping [11].

Authors [10] describe an appropriate algorithm for arm self-collision avoidance. This paper shows a complete algorithm, but it does not show integration with environmental collision avoidance.

Authors [7] describe a complete algorithmic solution but without implementation.

The aim of the research

Develop a method that will find the best grasping position and plan a robotic arm's c motion from a current position to the desired grasping position. After grasping is done, a gripper should hold the object with enough but not excessive force.

Main material

Grasping

One of the most significant areas of interaction between robot and environment is the manipulation of objects. Manipulation can be a part of many highly useful processes: assembling, sorting, moving, ordering, cargo loading etc. These processes are used in industrial and casual robots.

Robotic manipulators should have an end effector, also known as End of Arm Tooling [4]. Robot end-effectors are subdivided into next types:

- Impactive: physically grasp by applying direct force to the object.
- Ingressive: penetrate the surface of the object to pick it up.
- Astrictive: attractive forces are applied, like vacuum or magnetism.
- Contigutive: requires direct contact for adhesion.

The considered problem is related to the impactive type of gripper.

Fin-Ray effect

The Fin-Ray effect is an effect that describes a flexible construction that bends around an object when force toward the flexible structure is applied. Such construction gives an additional contact area that provides additional friction. Rays of fish fins inspired the structure. Fish fins have a structure with two bones that are attached to each other with elastic tissue. The tail fin is the main point to apply force for the movement. The tin consists of several basic structures stacked one above the other. The structure must be light but strong enough since excess weight would produce unnecessary energy loss. The design, which copies a fish fin, consists of two attached longitudinal fibers. There are cross fibers among the longitudinal fibers that keep the whole structure after assembly. The cross and longitudinal fibers are connected flexibly, which allows the required movement between them [5].

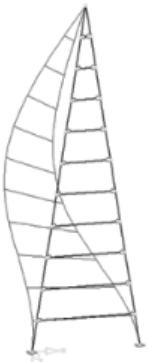

Fig.1. Fin-Ray effect

Finger material

The research team was faced with a choice of how to produce fingers for the gripper. 3D-printing was chosen as the most straightforward way which meets all requirements. The fingers are done with FDM 3D printing with TPU material. Other parts of the case are made with aluminum and ABS plastic.

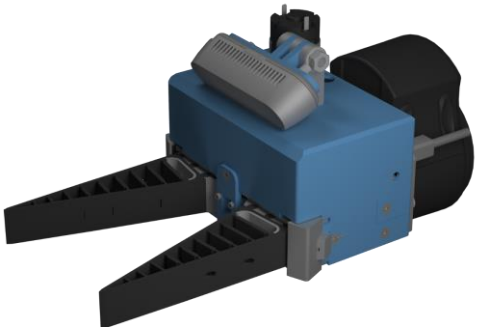

Fig. 2. 3D-model of the used gripper

Environment perception

Next, we must detect and store information about the environment. The system should scan the environment with perception input devices, represent it as a data structure and process to avoid obstacles during path planning.

Since the robotic arm can move in 3D space, it should be full of volumetric obstacle detection. Usually is done by binocular or monocular 3D scanning using SLAM [2; 3].

SLAM – simultaneous localization and mapping, is an approach to process multiple frames and sensor positions to build a volumeric model of the environment.

Visual slam can be performed even with a single camera (monocular) setup. It is cheap and straightforward. Depth is not fully available from one image; instead, we need multiple images to match them and compute a disparity.

With two cameras, it is even easier – we have a fixed distance between cameras. Having the disparity, we can set a distance from the camera for each point and, based on it, build a 3D representation of an image for the current frame.

The Intel RealSense depth camera D435 is a stereo solution, offering quality depth for various applications. It is a wide field of view that is perfect for applications such as robotics or augmented and virtual reality, where the most important part is seeing as much as possible. The Intel RealSense D435 depth camera has a wider FOV at approximately 85° field of view. D435 depth camera has a global shutter. Cameras with a rolling shutter record all the pixels in a scene by rapidly scanning either left and right or vertically. This will usually happen for a couple of frames, but the data will be saved as a single frame. Global shutter cameras operate differently in that they snapshot the whole scene in a single frame, so every pixel is captured simultaneously.

The whole SLAM approach implies building a full map. SLAM can use multiple different types of sensors, and the powers and limits of various types of sensors have been a significant driver of new algorithms. Statistical independence is required to cope with metric bias and noise in measurements. Data from different types of sensors can be processed by different SLAM algorithms whose approaches are more compatible with the sensors [6].

In our case, we have SLAM with odometry, lasers, and now, octrees from the point cloud.

Software platform

To organize the project was used ROS – Robot Operating System. It is big framework, even similar to the operating system as it called, besides it is not an operating system. ROS also provides some built-in packages, for navigaition, simulation, localization etc [1].

To test and debug anything without working with the real robot we can use Gazebo which simulates robot in virtual space.

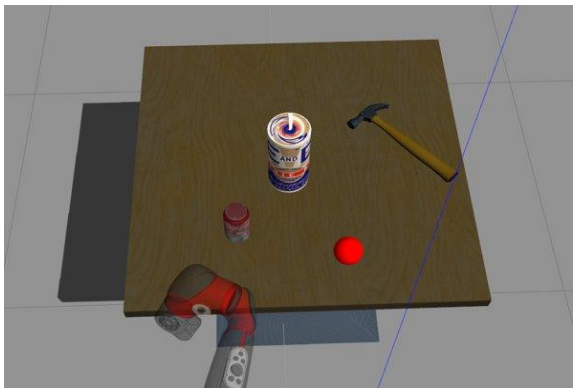

Fig. 3. Simulation in Gazebo

Point cloud

After the robot received any depth data from the image, it should store this data. One way is a point cloud – a set of points in space. When a camera (or it is generated with multiple frames) captured a depth image, each pixel of the depth image can be converted to a point in space relative to the camera position. Then this data could be added to previously generated point clouds from previous frames.

But if the robot only captures and adds the data, data will become bigger and bigger. Even when the camera sees the same object, it will add more and more points. It is an unnecessary big amount of data.

Voxels and Octree

One of the ways to represent a 3D environment discretely is voxels. A voxel is a volumetric pixel. Basically, a voxel is a value in the regular grid. Usually, the value is a flag if it is occupied; additionally, it can contain any data, like color. In the typical case, if we have a voxel grid, the voxel does not represent a real position in 3d space; it contains only its position instead. Such an approach can be more efficient than more common polygonal graphics, but sometimes it can be more computationally expensive.

But it also can be optimized – with octree. An octree is a tree-based data structure similar to voxels. It can be understood as variable size voxels.

The idea is to make one scene-size octant then reclusively divide it into eight octants if not all space inside the octant is occupied.

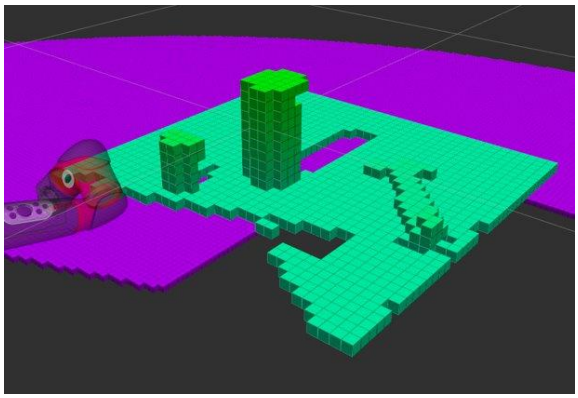

Fig. 4. Octree

The OctoMap library implements a 3D occupancy grid mapping approach by providing data structures and mapping algorithms in C++, especially suitable for robotics. In this research, I OctoMap as part of MoveIt! framework.

Grasp pose detection

Before grasping any object, the object should be recognized and chosen. It is a problem in robotics arm control to solve. Approaches to grasp perception can be subdivided into the following ways:

- With known models. Based on the recognition of predefined loaded models. The method tries to find similar objects among predefined CAD models. It can be more accurate with known objects but usually does not work if the object is not recognized.
- Without known models. When the system allows finding a grasping position for the object even if it is an unknown object, but at the same time, it can also support the detection of known objects.

There is an implementation of the second approach – the Grasp Pose Detection (GPD) package. It provides 6-DOF grasp poses for objects. It means that an object can be grasped from any side with any orientation if the arm can provide such a pose. At the moment, it supports only two-finger parallel grippers, which is appropriate for the gripper we use. Besides, the gripper is not a parallel jaw gripper, rather a scissors-type [11].

As an input, GPD consumes point cloud data and outputs available grasp poses.

The main strengths of GPD are:

- works for novel objects (no CAD models required for detection),
- works in dense clutter, and
- outputs 6-DOF grasp poses (enabling more than just top-down grasps).

Arm control with MoveIt

MoveIt is an easy-to-use open-source robot manipulation platform for commercial application development, prototyping, and testing algorithms [9].

MoveIt has a plugin interface to work with motion planners. It allows MoveIt to use and communicate with different motion planners which can be provided by multiple libraries. That makes MoveIt easily extensible [8].

To represent the actual world around the robot and store its and robot's state the framework uses a planning scene. It is supported by the planning scene monitor inside the move group node. The planning scene monitor subscribes to:

- State Information: on the joint_states topic;
- Sensor Information: use the world geometry monitor described below;
- World geometry information: gets data from user input on the planning_scene topic (as a planning scene diff).

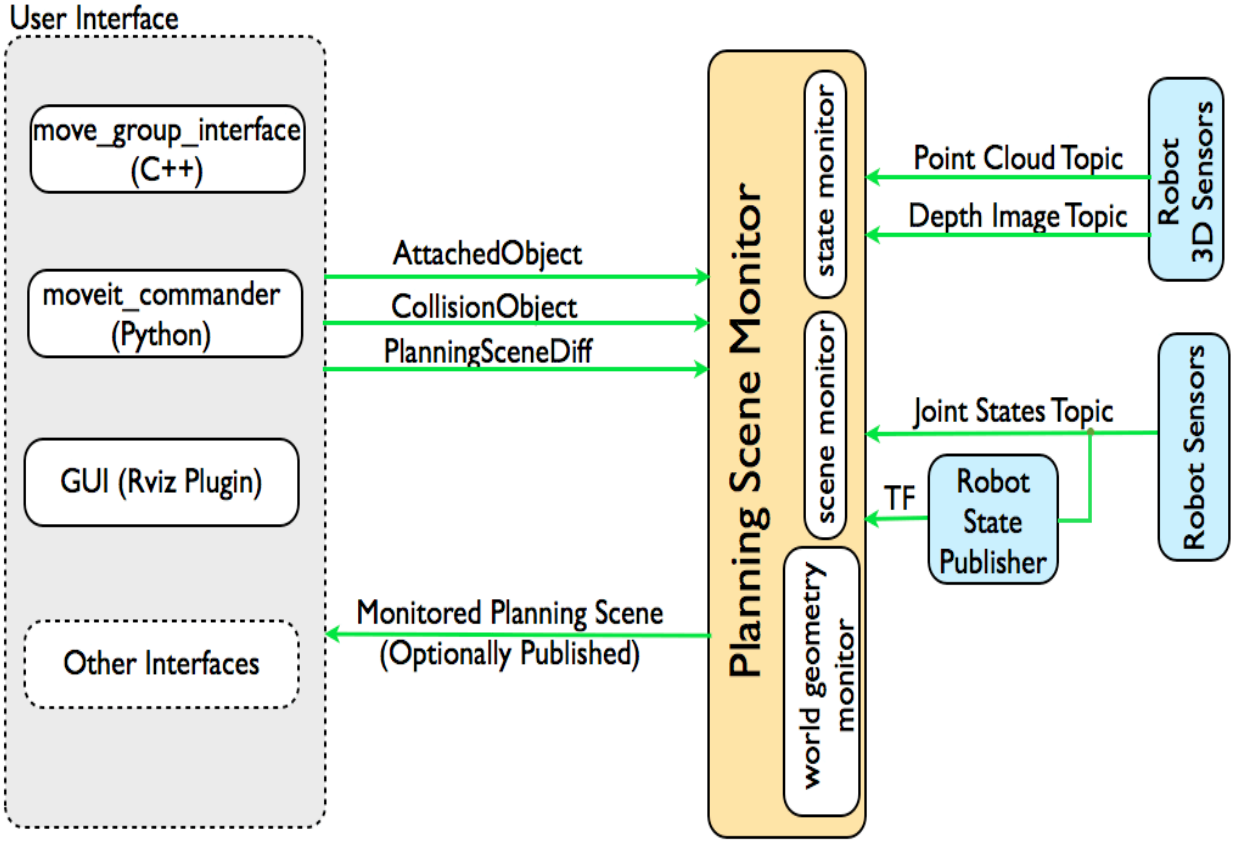

Fig. 5. Planning scene structure

The Occupancy map monitor is developed by using Octomap to represent the occupancy map of the environment. Octomap can encode probabilistic information about individual cells, but this information is not used in current MoveIt version. The Octomap output as octree can directly be passed into Flexible Collision Library, the collision checking library that MoveIt uses.

Collision checking in MoveIt is configured inside a Planning Scene with the CollisionWorld object. Fortunately, MoveIt is set up so that end users never really have to worry about how collision checking is happening internally. Collision checking in MoveIt is mainly processed using the FCL package – MoveIt's primary collision checking library.

MoveIt supports collision checking for different types of objects, including: (1) Meshes; (2) Primitive Shapes – e.g. boxes, cylinders, cones, spheres, and planes; (3) Octomap – octree generated by Octomap can be directly used for collision checking.

Slip detection

The easiest way to achieve haptic perception is to use an acceleration sensor. This can be used to record vibrations on the gripper in the X, Y and Z directions. The acceleration sensor should be mounted as far in front of the effector as possible in order to avoid damping of the vibration by the elastic TPU.

In addition, the damping property could thereby be used to reduce vibrations from outside [7].

Another possibility is to record the forces on the connection flange of the gripper. This also enables vibrations on the gripper to be recorded in the X, Y and Z directions. For this purpose, strain gauges are glued to the flange of the gripper. Due to the changes in resistance, the vibrations that act on the flange can be calculated back. In addition, conclusions can be drawn about the weight of the gripped object. The elastic TPU may dampen the vibration on the finger so much that it disappears in the noise.

Direct measurement of the gripped object is conceivable with an optical mouse sensor or a trackball/trackpad fitted in the recess. This would make it possible to detect an object slipping out directly. The detour via a time-consuming and computationally intensive evaluation could perhaps be omitted.

Conclusions

As a result of the work, the complete solution was developed. This solution provides a full pipeline for detection, motion planning, and grasping.

During the research, existing algorithms and solutions were considered. After the full analysis of the problem and available solutions, a complete system that solves the given problem was presented.

Література

1. Koubaa A. (2016). *Robot Operating System (Volume 1).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-26054-9.
2. Koubaa A. (2017). *Robot Operating System (Volume 2).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-54927-9.
3. Koubaa A. (2019). *Robot Operating System (Volume 3).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-91590-6.
4. Fairchild C., Harman T.L. (2017). *ROS Robotics By Example - Second Edition: Learning. to control wheeled, limbed, and flying robots using ROS Kinetic Kame.* Packt Publishing, 428.
5. Pfaff O., Simeonov S., Cirovic I., Stano, P. (2011). *Application of finray effect approach for production process automation*. Vienna, Austria: DAAAM International, 1247-1248.
6. Кошель С. О., Ковалёв Ю., Манойленко О. П. (2019). *Проектування промислових роботів та маніпуляторів.* Центр навчальної літератури, 256.
7. Платонов А.К., Соколов С.М., Трифонов О.В. (2017). *Алгоритмы управления движением схвата манипулятора*. Москва: ИПМ им. М.В.Келдыша РАН. DOI: 10.20948/prepr-2017-47.
8. Safeea M., Neto P., Bearee R. (2019). *On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case.* Robotics and Autonomous Systems, Elsevier, 278-288.
9. Kivelä T., Mattila J., Puura J. (2018). *Redundant Robotic Manipulator Path Planning for Real-Time Obstacle and Self-Collision Avoidance.* Mechan. Machine Science, vol. 49, Springer. DOI: 10.1007/978-3-319-61276-8_24.
10. De Santis A., Albu–Sch¨affer A., Ott Ch., Siciliano B., Hirzinger G. (2007). *The skeleton algorithm for self-collision avoidance of a humanoid manipulator The skeleton algorithm for self-collision avoidance of a humanoid manipulator*. IEEE/ASME international conference on advanced intelligent mechatronics, Zurich, 1-6. DOI: 10.1109/AIM.2007.4412606.
11. ten Pas A, Gualtieri M, Saenko K, Platt R. (2017). *Grasp Pose Detection in Point Clouds.* The International Journal of Robotics Research, Vol 36, Issue 13-14, 1455-1473. DOI: 10.1177/0278364917735594

References

1. Koubaa A. (2016). *Robot Operating System (Volume 1).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-26054-9.
2. Koubaa A. (2017). *Robot Operating System (Volume 2).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-54927-9.
3. Koubaa A. (2019). *Robot Operating System (Volume 3).* Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-91590-6.
4. Fairchild C., Harman T.L. (2017). *ROS Robotics By Example - Second Edition: Learning. to control wheeled, limbed, and flying robots using ROS Kinetic Kame.* Packt Publishing, 428.
5. Pfaff O., Simeonov S., Cirovic I., Stano, P. (2011). *Application of finray effect approach for production process automation*. Vienna, Austria: DAAAM International, 1247-1248.
6. Koshel S. O., Kovalyov Yu., Manoilenko O. P. (2019). *Proektuvannia promyslovykh robotiv ta manipuliatoriv*. Tsentr navchalnoi literatury, 256. (in Ukrainian)
7. Platonov A.K., Sokolov S.M., Tryfonov O.V. (2017). *Alhorytmy upravlenyia dvyzhenyem skhvata manypuliatora*. Moskva: YPM ym. M.V.Keldysha RAN. DOI: 10.20948/prepr-2017-47. (in Russinan)
8. Safeea M., Neto P., Bearee R. (2019). *On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case.* Robotics and Autonomous Systems, Elsevier, 278-288.
9. Kivelä T., Mattila J., Puura J. (2018). *Redundant Robotic Manipulator Path Planning for Real-Time Obstacle and Self-Collision Avoidance.* Mechan. Machine Science, vol. 49, Springer. DOI: 10.1007/978-3-319-61276-8_24.
10. De Santis A., Albu–Sch¨affer A., Ott Ch., Siciliano B., Hirzinger G. (2007). *The skeleton algorithm for self-collision avoidance of a humanoid manipulator The skeleton algorithm for self-collision avoidance of a humanoid manipulator*. IEEE/ASME international conference on advanced intelligent mechatronics, Zurich, 1-6. DOI: 10.1109/AIM.2007.4412606.
11. ten Pas A, Gualtieri M, Saenko K, Platt R. (2017). *Grasp Pose Detection in Point Clouds.* The International Journal of Robotics Research, Vol 36, Issue 13-14, 1455-1473. DOI: 10.1177/0278364917735594

Received 15.06.2020
Accepted 17.08.2020